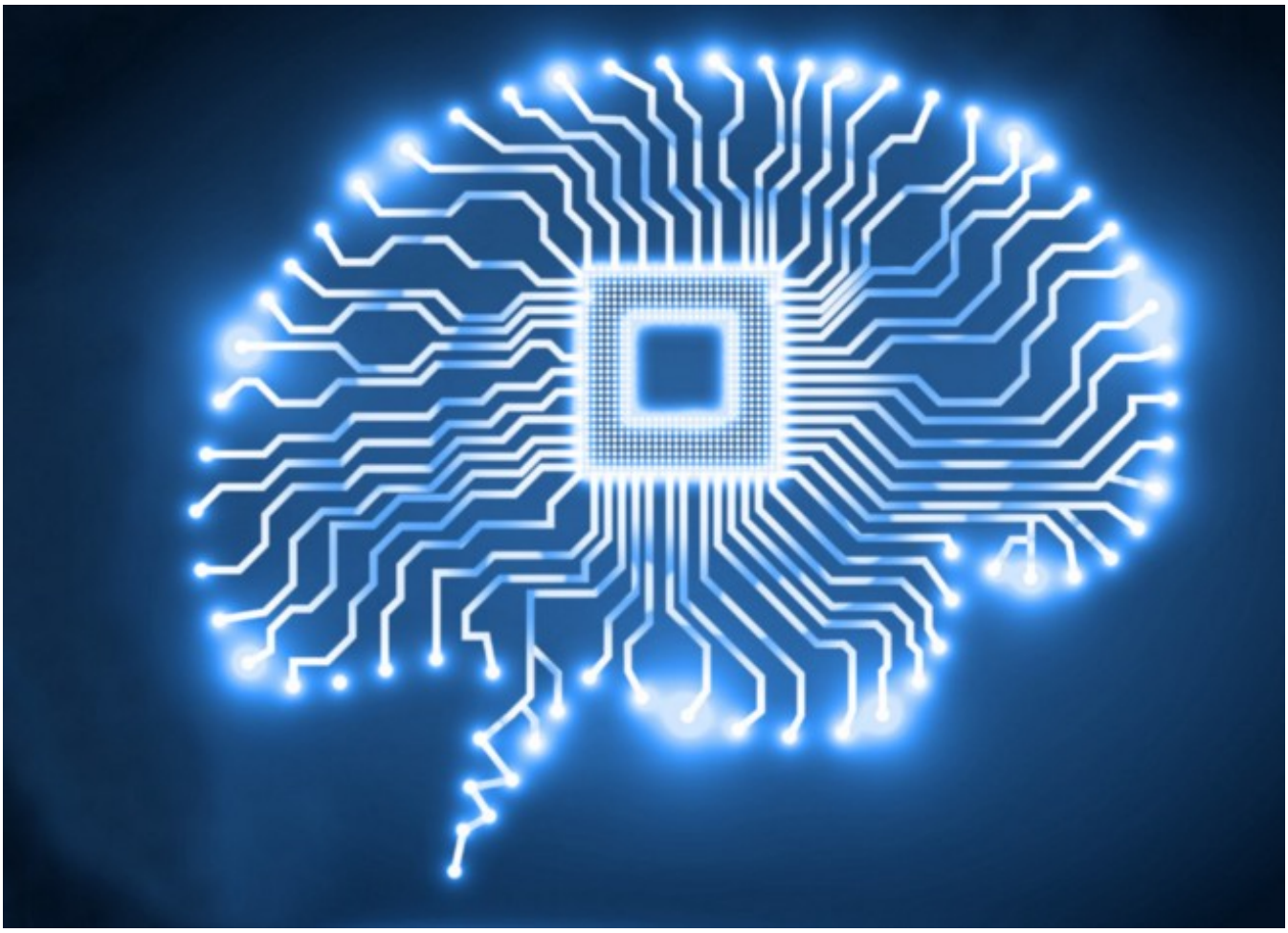
Marlon Sousa

4 Followers About

Detecção de Phishing por NLP — Natural Language Processing

Marlon Sousa Sep 8, 2020 · 3 min read



NLP é um campo do Machine Learning que faz o computador aprender, identificar, analisar e até gerar linguagem humana.

```
[1]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
```

As bibliotecas matplotlib e seaborn não serão usadas aqui

Dados

Os dados usados: https://www.kaggle.com/taruntiwarihp/phishing-site-urls

Após download dos dados faremos a leitura usando o pandas

```
[2]: df = pd.read_csv('phishing.csv')

[3]: df.head()

[3]:
```

	URL	Label
0	nobell.it/70ffb52d079109dca5664cce6f317373782/...	bad
1	www.dghjdgf.com/paypal.co.uk/cycgi-bin/webscrc...	bad
2	serviciosbys.com/paypal.cgi.bin.get-into.herf....	bad
3	mail.printakid.com/www.online.americanexpress....	bad
4	thewhiskeydregs.com/wp-content/themes/widescre...	bad

Tratamento dos Dados

Você pode fazer o tratamento da coluna 'URL' usando a biblioteca string, nesse caso não irei usar pois faria um replace de símbolos na URL e poderia danificar nosso resultado final.

```
[6]: import string

[7]: ponto = string.punctuation
     ponto

[7]: '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

Separação dos Dados

Nesta etapa faremos a separação dos dados em parte de treino e teste.

```
[65]: x = df['URL']
      y = df['Label']

[66]: cv = CountVectorizer()
      x = cv.fit_transform(x)

[67]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=101)
```

O CountVectorizer faz a Vetorização dos dados.

Vetorização é o processo de codificação de texto como números inteiros, ou seja, forma numérica para criar vetores de recursos para que algoritmos de aprendizado de máquina possam entender nossos dados.

Naive Bayes

Aqui faremos uso do naive_bayes quem vem no sklearn para Classificação Binária.

```
[68]: from sklearn.naive_bayes import MultinomialNB

[69]: nb = MultinomialNB()
```

Treino

Após iniciarmos a instância de MultinomialNB podemos fazer o treino com base em nossos dados que separamos para treino.

```
[70]: nb.fit(x_train, y_train)

[70]: MultinomialNB()
```

Predict

Após fazermos o treino podemos fazer uma predição utilizando os dados de teste.

```
[71]: pred = nb.predict(x_test)

[72]: from sklearn.metrics import accuracy_score

[73]: print('Accuracy: {:.2f}%' .format(accuracy_score(y_test, pred)*100))
      Accuracy: 96.38%
```

Matemática

Naive Bayes

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Probabilidade de y dados recursos de entrada X.

$$P(y|X) = \frac{P(X|y) * P(y)}{P(X)}$$

$$P(X|y) = P(x_1|y) * P(x_2|y) * ... * P(x_n|y)$$

$$P(y|X) \propto P(X|y) * P(y)$$

-or-

$$P(y|X) \propto P(y) * \prod_{i=1}^{n} P(x_i|y)$$

$$y = argmax_y[P(y) * \prod_{i=1}^{n} P(x_i|y)]$$

Argmax é simplesmente uma operação que encontra o argumento que fornece o valor máximo de uma função de destino. Nesse caso, queremos encontrar o valor máximo de y.

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_i)^2}{2\sigma_y^2}\right)$$

NLP Machine Learning Python